(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－151743
（Ｐ２０００－１５１７４３Ａ）
(43)公開日　平成12年５月30日(2000.5.30)

(51)Int.Cl.[7]	識別記号	Ｆ Ｉ			テーマコード（参考）
H04L 29/06		H04L 13/00	305	Z	5B076
G06F 9/06	530	G06F 9/06	530	W	5K034

審査請求　未請求　請求項の数６　ＯＬ　（全17頁）

(21)出願番号　特願平10－328053

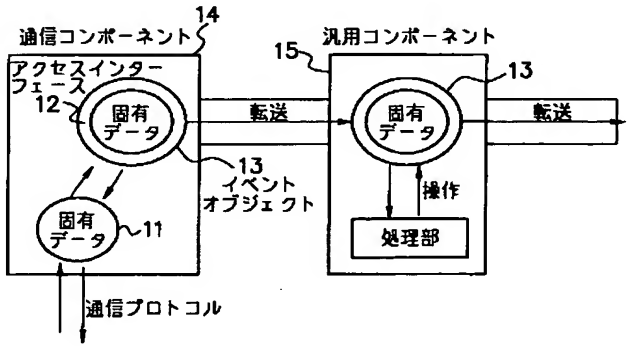
(22)出願日　平成10年11月18日(1998.11.18)

(71)出願人　000004226
日本電信電話株式会社
東京都千代田区大手町二丁目３番１号
(72)発明者　依田　育生
東京都新宿区西新宿３丁目19番２号　日本電信電話株式会社内
(72)発明者　川幡　太一
東京都新宿区西新宿３丁目19番２号　日本電信電話株式会社内
(74)代理人　100069981
弁理士　吉田　精孝

最終頁に続く

(54)【発明の名称】ネットワーク制御システムの設計方法

(57)【要約】
【課題】　プロトコルの多様化に対応し得るコンポーネントウェアの組み合わせによるネットワーク制御システムの設計方法を提供すること。
【解決手段】　プロトコル固有の固有データ１１にプロトコルによらないアクセスインターフェース１２を実装させてイベントオブジェクト１３を構成することによって、プロトコル固有のデータ構造を保持させたままコンポーネント１４，１５間で交換可能とし、プロトコルによらない設計を可能とする。



1

【特許請求の範囲】

【請求項１】　ネットワーク管理システムであって、ネットワークの通信装置を制御する機能要素となるコンポーネントウェアを用意・選択して、選択されたコンポーネントウェアの機能を使用する目的に合わせてカスタマイズし、このように設定されたコンポーネントウェアを幾つも接続することで設計を行うネットワーク制御システム設計方法において、

接続されたコンポーネントウェア間で、多様なデータ構造のデータ自体は異なりつつも、統一されたデータアクセスインターフェースでアクセスを可能にすることによって、コンポーネントウェアで扱うデータまたはコンポーネントウェア自身でカスタマイズするデータまたはコンポーネントウェア間でやりとりされるのをモニタするデータに対する表示及び編集の方法を統一することを特徴とするネットワーク制御システムの設計方法。

【請求項２】　２つのコンポーネントウェア間を接続させる操作をする時に、データを出力する側のコンポーネントウェアの出力制約条件が、入力側の制約条件と同じか、または厳しい場合は接続を許すようにコンポーネントウェアを誘導し、そうでない場合はそれらのコンポーネントウェア間の接続を拒絶する、ことを特徴とする請求項１記載のネットワーク制御システムの設計方法。

【請求項３】　設計時に２つのコンポーネントウェア間を接続する際、コンポーネントウェア間でデータを試験的にやりとりし、データ出力側のコンポーネントウェアから出力されるデータに対して制約情報を取得し、この取得したデータの制約条件によって、データ出力側のコンポーネントウェア自身の出力制約情報をより厳しくし、対象のコンポーネントウェアと接続できるようにするツールを用意したことを特徴とする請求項１記載のネットワーク制御システムの設計方法。

【請求項４】　ネットワーク管理システムにおける管理対象となる機器のｈｔｔｐサーバが出力するＨＴＭＬファイルを、機器を管理するシステムがトラップする手段と、

そのトラップしたＨＴＭＬファイルを、機器の現状における接続・設定の状態に照らして一部改変または最初から初期設定値を追加する等の処理を行うＨＴＭＬ処理手段と、

処理後のＨＴＭＬを、そのネットワークシステムの管理者用のブラウザに転送する転送手段とを有することを特徴とする請求項１記載のネットワーク制御システムの設計方法。

【請求項５】　コンポーネントウェアまたは通信プロトコルの都合によって、設計時及び運用時に、本来１つのデータオブジェクトとして処理すべきデータを、適切に分割・走査して、個々のデータを適切なコンポーネントウェアに渡して処理をさせることを特徴とする請求項１乃至４いずれか記載のネットワーク制御システムの設計方法。

2

【請求項６】　設計時に接続するコンポーネントウェア群の中に、運用時にはインスタンスが１つしか存在しないコンポーネントウェア群と複数個存在するコンポーネントウェア群との分界点となる分界点コンポーネントウェアがあり、

設計時のシミュレートを行う際に、この分界点コンポーネントウェアを境に、分界点コンポーネントウェア自身が、分界点のコンポーネントブロックのインスタンスを多数作成・初期化し、

分界点コンポーネントウェアが複数個側へ適切にデータの流れを振り分け処理し、

ネットワークにおける１：ｎの関係をシミュレートするツールを用意することを特徴とする請求項１乃至５いずれか記載のネットワーク制御システムの設計方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、ネットワークを構成する各種の通信装置（ネットワークエレメント：ＮＥ）を保守・運用・管理するネットワーク制御システムの設計方法、特にコンポーネント指向を用いたネットワーク制御システムの設計方法に関するものである。

【０００２】

【従来の技術】ソフトウェアは一から作る時代から、組み合わせる時代に変わりつつある（青山幹雄：「ソフトウェアビッグバン」、情報処理、３９巻４号、ｐｐ．３３８～３４１、１９９８）。通信網管理システムでも、コンパイルせずにそのまま利用できるプラグ＆プレイ型ソフトウェア部品（コンポーネントウェア）を用意し、その組み合わせでシステムを構築することができれば、開発期間、コスト共に減少させることができる。ＴｅｌｅＭａｎａｇｅｍｅｎｔ　ＦｏｒｕｍにおけるＳＭＡＲＴ　ＴＭＮ（ＴｅｌｅＭａｎａｇｅｍｅｎｔ　Ｆｏｒｕｍ　ＳＭＡＲＴＴＭＮ　Ｗｏｒｋ　Ｐｒｏｊｅｃｔ、ｈｔｔｐ://www.tmforum.org/pages/smart.html）等でもその方向を目指しているが、コーディングレスに達するまでにはまだ技術的に解決しなければならない課題が多い。

【０００３】ＣＯＲＢＡ（ＯＭＧ　ＩＤＬ　Ｖｅｒ．２．２　Ｆｅｂ－１９９８）、ＤＣＯＭ（古山一夫「ＤＣＯＭガイドブック」、オーム社、１９９７）や、ＪａｖａＢｅａｎｓ（ＪａｖａＢｅａｎｓ™ ＡＰＩ ｓｐｅｃｉｆｉｃａｔｉｏｎ １．０１ ＳＵＮ Ｍｉｃｒｏｓｙｓｔｅｍｓ，２４－Ｊｕｌ－１９９７）等の有力な枠組みが業界標準となり、インターフェースリポジトリやイントロスペクション等の仕様獲得機能が限定的ながら利用可能になったことにより、異なる組織で作られたコンポーネントの結合や、ツール製作者とコンポーネント製作者との分業が可能になってきている。

【０００４】ネットワーク制御システム（ＯｐＳ）は通信ネットワークを構成する各種の通信装置を保守・運用

・管理する。このＯｐＳの設計方法として、コンポーネントウェアを用いる方法が、依田育生・矢田浩二「コンポーネント指向通信網管理システム」（１９９８年電子情報通信学会ソサイエティ大会予稿集Ｂ－１４－１８、ｐｐ．５６５、（社）電子情報通信学会、１９９８．９．７発行）、川幡太一・依田育生「通信網管理システムデータ変換部品の設計と実装」（１９９８年電子情報通信学会ソサイエティ大会予稿集Ｂ－１４－１９、ｐｐ．５６６、（社）電子情報通信学会、１９９８．９．７発行）、前大道浩之・後藤哲明・多胡誠久「通信網管理システム用プロトコル部品の設計と実装」（１９９８年電子情報通信学会ソサイエティ大会予稿集Ｂ－１４－２０、ｐｐ．５６７、（社）電子情報通信学会、１９９８．９．７発行）、川幡太一・矢田浩二「コンポーネント指向ＯｐＳのプロトタイプ」（１９９８年電子情報通信学会ソサイエティ大会予稿集Ｂ－１４－２１、ｐｐ．５６８、（社）電子情報通信学会、１９９８．９．７発行）で提案されている。

【０００５】図１はＯｐＳの代表的な構成の概要を示すもので、同図(a)は階層型のシステム、同図(b)はコンポーネント型のシステムをそれぞれ示している。

【０００６】コンポーネント型のシステムの設計には、コンポーネントウェア（ＣＷ）と呼ばれる部品が用いられる。

【０００７】図２は従来のコンポーネント型のＯｐＳの設計の流れを示すもので、パレット１上に事前に複数種類の汎用的なコンポーネントウェア（コンポーネント）２が用意されており、これらのうち、実際にＯｐＳを作成するのに必要なコンポーネント２を選択し、該選択したコンポーネント２を適宜、カスタマイズし、それらを接続することによって作成していた。

【０００８】この方法はまた、プロトコルの差異をコンポーネント化することにより、プロトコルの多様化の問題も解決した。

【０００９】図３は前述したＯｐＳの設計におけるデータの設定のようすを示すもので、コンポーネント２における各種のデータの設定はプロパティシート３と呼ばれる設定領域を開き、そこに各データ固有の表現方法で値を記述することにより行われる。また、各コンポーネント２間でやりとりされるデータを表示させ、モニタ（４）することもできるが、この表示も各データ固有の表現方法で行われる。

【００１０】このようなＯｐＳでは、図４に示すように、ユーザ５はアプリケーション６を介してネットワークエレメント（ＮＥ）７を操作する。ネットワークエレメント７を管理するアプリケーション６は、幾つものコンポーネント２の接続で構成され、ユーザはＧＵＩコンポーネントであるボタン８及びグラフ表示９によって操作（ＧＵＩ操作）する如くなっている。

【００１１】ネットワークエレメント７からのイベントは、アプリケーション６内のコンポーネント２間でやりとりして処理され、グラフ表示９を通してユーザへ伝えられ、また、ユーザの指示はボタン８を通してネットワークエレメント７に伝えられる。

【００１２】

【発明が解決しようとする課題】（ａ）コンポーネント間でやりとりされる構造型データへの統一的なインターフェース

ＣＭＩＰ、ＳＮＭＰ等の伝統的な管理プロトコル、ＣＯＲＢＡ等の分散の枠組みに加え、ＨＴＭＬ＋Ａｐｐｌｅｔによるユーザインターフェース等、ＯｐＳが扱うプロトコルは多様化している。さらに、ルータ等のネットワーク機器では、詳細な設定はＨＴＭＬを用い、ブラウザ経由で行うものが多くなっており、これらは単体で管理する上では便利であるが、多数のネットワーク機器を集中的に管理したい場合、ＨＴＴＰを新たな管理プロトコルとして捉え、機械処理する必要が生じている。

【００１３】これらの多様な管理プロトコルを扱う場合、これまでは各管理プロトコルのプロトコルデータユニット（ＰＤＵ）の構造を意識しながら、プログラムを製作する必要があった。例えば、ＣＭＩＰ、ＳＮＭＰではＡＳＮ．１を、ＣＯＲＢＡではＩＤＬを用いてデータが記述され、符号化方法や計算機言語へのマッピングもそれぞれ異なる。これらのプロトコル毎に専用のコンポーネント群を用意していくと、数多くのコンポーネントを設計しなければならないだけでなく、開発者にとっても考え方の異なる多くのコンポーネントを扱う必要が生じ、負担が増してしまう。

【００１４】管理プロトコルを計算機言語で実装する際には、最も効率的に処理できるようにＰＤＵの構造と計算機言語上のデータ構造のマッピングが図られているのが常識であり、効率の面からは既存のプロコルスタックで用いられているデータ構造を尊重するべきである。異なるプロトコルでコンポーネントを共有化するために、コンポーネント間で受け渡すデータを、安易に共通なデータ構造に変換すると、場合によってはプロトコルの処理に必須な情報が欠落して通信不能になる可能性がある。

【００１５】コンポーネントウェアを指向したネットワーク制御システムの設計方法では、デバッグや値の入力、コンポーネントのプロパティシートにおける値の設定が非常に面倒であったため、これらを改良する必要が認められた。

【００１６】（ｂ）集中ネットワーク管理における管理対象機器の持つＧＵＩインターフェースの利用

今後、ネットワーク管理で使用されるユーザーインターフェースは、開発コストがかさむアプリケーションではなく、安価なｈｔｔｐ＋ａｐｐｌｅｔが主流になると思われる。また、管理対象となる機器も、ｈｔｔｐによる管理手法を提供するようになっている。そのため、多数

のネットワーク機器の一括管理はこれまでの方法で行うとしても、個々の機器を管理する際には、それらの機器が提供する管理手法を最大限活用することが望ましい。

【００１７】（ｃ）データタイプと誘導

ソフトウェア部品の再利用を困難にしている原因の１つは、部品間で引き渡すデータタイプの不一致である（Ｇｏｒｄｏｎ　Ｓ．Ｎｏｖａｋ　Ｊｒ．”Ｓｏｆｔｗａｒｅ　Ｒｅｕｓｅ　ｂｙ　Ｓｐｅｃｉａｌｉｚａｔｉｏｎ　ｏｆ　Ｇｅｎｅｒｉｃ　Ｐｒｏｃｅｄｕｒｅｓ　ｔｈｒｏｕｇｈ　Ｖｉｅｗｓ”，ＩＥＥＥＴｒａｎｓ　ｏｎ　Ｓｏｆｔｗａｒｅ　Ｅｎｇｉｎｅｅｒｉｎｇ，ｖｏｌ．２３，Ｎｏ．７，ｐｐ．４０１～４１７，Ｊｕｌｙ，１９９７）。

【００１８】一方で、コンパイラ等によって検査される強力な型付けは、開発者を適切に誘導し、プログラムの誤りを防ぐ有力な手段となっている（ＮＭＦ０４０，４１，４２，４３　ＴＭＮ／Ｃ++ＡＰＩ，１９９７）。通信網管理で扱うプロトコルのデータ構造は複雑になる場合が多く、このような誘導は必須である。

【００１９】ところが、このデータタイプは情報定義毎に異なるのが一般的で、任意に結合可能な、汎用のコンポーネントを設計しようとすると、型付けを廃してデータタイプの抽象化を図る必要が生じる。これでは実行時にデータが渡せるかどうかを開発者が自分で調べることになり、ますます開発者の負担が増えるため、型付けに代わる誘導の仕組みが必須となる。

【００２０】（ｄ）コンポーネント間でやりとりされるデータの走査

従来技術では、コンポーネント間でやりとりされるデータは、一括してまとめて受け渡しを行っていた。そのため、ＨＴＭＬデータや、ＣＭＩＰのＬｉｎｋｅｄ　Ｒｅｐｌｙ、ＡＳＮ．１のｓｅｔ　ｏｆ等のように、全体としては１つのデータだが、個別の要素に分解して個別に処理をしたい場合には、従来の手法では対応が困難だった。

【００２１】（ｅ）ネットワーク上における、１：ｎの関係をコンポーネントで構築する際の問題

従来のコンポーネント技術の利点の１つとして、コンポーネントをツールで組立てながら動的に実行して動作を確認することが可能なことが挙げられる。しかし、この技術をＯｐＳ構築に適用しようとすると、ＯｐＳの中でしばしば現れる、マネージャ・エージェント、またはサーバ・クライアントのような１：ｎの関係では、ｎの方の状態が不特定多数個出現する。そのため、このような関係をツール上で動的にシミュレートすることが困難であった。

【００２２】

【課題を解決するための手段】（ａ）コンポーネント間でやりとりされる構造型データへの統一的なインターフェース

実際にコンポーネント間でやりとりされる様々なデータ（ＡＳＮ．１やＸＭＬ等の多様なデータ構造）についで調べて、コンポーネント間でやりとりされる構造型データに対して統一的なツリー構造のインターフェースを定めた。これによって、データ自体は異なりつつも、統一されたツリー型データアクセスインターフェースでアクセスできるようにする。例えば、管理システムで使われるＣＷの各種設定をツリー構造メニューで行うことが可能になったり、コンポーネント間でやりとりされるデータのモニタをする際、そのデータを統一された形式で分かり易く見ることができるようになった。

【００２３】（ｂ）集中ネットワーク管理における管理対象機器の持つＧＵＩインターフェースの利用

個々のネットワーク機器を接続した際に、機器のｈｔｔｐサーバが出力するＨＴＭＬファイルを、機器を管理するシステムが管理データとして取り込む。このデータに対して、機器の現状における接続・設定の状態に照らして、一部分改変、または最初から初期設定値を追加する等の処理を行う。その処理後、データを再びＨＴＭＬに直して、そのネットワークシステムの管理者用のブラウザに転送する。

【００２４】（ｃ）データタイプと誘導

コンポーネント間での接続を適切に誘導するために、コンポーネントがやりとりできるデータに対して制約を導入し、この制約に基づいて、コンポーネント間の接続の可否（誘導）を決めることにする。この接続の誘導には、以下の２つの方法がある。

【００２５】（１）各コンポーネントから、出力・入力データの型情報とは別に、型が取りうる値についての情報（制約情報）を取得する。そして、データ出力側コンポーネントの出力制約条件が、データ入力側コンポーネントの入力制約条件に比べて同じか、または厳しい場合に限って、接続を許すようにコンポーネントを誘導する。

【００２６】（２）コンポーネントだけではなく、コンポーネントから入出力されるデータに対しても、制約情報を取得できるようにする。この場合、出力側コンポーネントから出力されるデータの制約条件は、実際に、そのコンポーネントが動くまで分からない。このため、コンポーネントをツールで組立てている途中で、実際にデータを出力するコンポーネントまでとりあえずデータを流す。この流れてくるデータの制約条件によって、コンポーネント自身の出力制約情報をより厳しくし、入力側コンポーネントと接続できるようにする。即ち、この手法は「組立てながらシミュレートする」ことができなければ意味がない。

【００２７】また、コンポーネント同士の接続判定を行う場合、データ出力コンポーネントの持つ出力制約条件が、データ入力コンポーネントの持つ入力制約条件に比べて厳しすぎるため、つながらないにも関わらず、実際

に流れてくるデータは、出力コンポーネント自身の出力制約条件よりも厳しいというような場合がある。この時、見かけはコンポーネント同士の接続は無理でも、事実上はコンポーネント同士の接続が可能となる。ここで、実際にコンポーネント自身の接続を可能にするには、出力コンポーネントのみから出力制約情報を得るのではなく、出力コンポーネントから流れてくる出力データに対しても、出力制約情報を得る必要がある。

【００２８】（ｄ）コンポーネント間でやりとりされるデータの走査

巨大なデータを、複数のより細かいデータとして走査するようなコンポーネントを採用する。その際、走査して流す個々のデータに対して何らかのコンポーネントが処理を行った結果を、再び元の巨大データに反映させるコンポーネントや、そうしないコンポーネント等を用意する。

【００２９】（ｅ）ネットワーク上における、１：ｎの関係をコンポーネントで構築する際の問題

１：ｎの分界点におけるコンポーネントを境にして、分界点からｎ側のコンポーネントブロックのインスタンスを、分界点コンポーネントがシミュレート時に、多数作成・初期化する。また、サーバまたはマネージャ側のデータをクライアントまたはエージェント側に適切に振り分けたり、その逆を行う処理も分界点コンポーネントが行う。このようにして、ツール上でもネットワークの１：ｎ関係を適切にシミュレートすることを可能にする。

【００３０】

【発明の実施の形態】

【００３１】

【実施の形態１】請求項１、２及び３に記載された発明が解決しようとする課題をまとめると、通信網管理のアプリケーションドメインにおけるソフトウェアのコンポーネント化を進めるに当たっては、コンポーネント間のデータ交換に対して以下が必要である。

【００３２】１）今後現れるより新しい管理プロトコルのＰＤＵ構造も包含できるような抽象的なデータ構造の設計

２）抽象化によって失われる型付けに代わる新たな開発者誘導の仕組みの創出

３）プロトコル固有の実装の尊重

本実施の形態では、分散機能を持たない細粒度のコンポーネントを対象とすることを前提としている。即ち、各コンポーネントは同一プロセス内部で動作し、コンポーネント間のデータの交換はプロシージャ呼び出しの引数として渡すものとする。

【００３３】同一プロセス内で結合するコンポーネント間で交換するデータは、一定の機能を持ったオブジェクト（イベントオブジェクト）とすることができる。上記の条件を満たすために、イベントオブジェクトには各管理プロトコル固有のデータ構造をそのまま保持させると共に、プロトコルによらないアクセスインターフェースを実装させる。

【００３４】図５は本発明のアクセスインターフェースの概要を示すもので、プロトコル固有の固有データ１１にプロトコルによらないアクセスインターフェース１２を実装させてイベントオブジェクト１３を構成することによって、プロトコル固有のデータ構造を保持させたままコンポーネント１４，１５間で交換可能としている。

【００３５】これ以降は、本発明のアクセスインターフェースの設計と実装の一例について述べる。

１．アクセスインターフェース

１．１．ＰＤＵデータ構造の整理

通信網管理で使用する情報定義手法としては、ＧＤＭＯ／ＡＳＮ．１（ＣＭＩＰ），ＭＩＢ／ＡＳＮ．１（ＳＮＭＰ），ＩＤＬ（ＣＯＲＢＡ），ＤＴＤ：Ｄｏｃｕｍｅｎｔ ｔｙｐｅ Ｄｅｃｌａｒａｔｉｏｎ（ＨＴＭＬ，ＳＧＭＬ，ＸＭＬ《Ｅｘｔｅｎｓｉｂｌｅ Ｍａｒｋｕｐ Ｌａｎｇｕａｇｅ（ＸＭＬ）１．０Ｗ３Ｃ Ｒｅｃ．１０－Ｆｅｂ－１９９８．http://www.w3.org/TR/REC-xml》、ＳＱＬ（ＲＤＢＭＳ）等が考えられる。イベントオブジェクトの持つインタフェースは、これらの手法を用いて表現できるデータ構造の全てを操作できなければならない。また、可能な限り単純であることが望ましい。

【００３６】まず、データ構造は基本型と、それを組み合わせた複合型との２つに大別できる。基本型は整数型や文字列型等、通常の計算機言語の原子型で表せるような型である。各プロトコルの定義手法において、基本型にはあまり差がない。基本型は計算機言語で直接的に表すことが可能であるため、アクセスインターフェースとしては、原子型毎に変更するメソッドと取得するメソッドを用意しておけば良い。

【００３７】これに対して複合型は、各プロトコルで異なる表現方法が用いられている。本発明では、オブジェクト指向設計におけるオブジェクトの集合を表す、Ｃｏｌｌｅｃｔｉｏｎ（Ｄｅｎｎｉｓ ｄｅ Ｃｈａｍｐｅａｕｘ，Ｄｏｕｇｌａｓ Ｌｅａ，ａｎｄ Ｐｅｎｅｌｏｐｅ Ｆａｕｒｅ，”Ｏｂｊｅｃｔ－Ｏｒｉｅｎｔｅｄ Ｓｙｓｔｅｍ Ｄｅｖｅｌｏｐｍｅｎｔ”，Ａｄｄｉｓｏｎ Ｗｅｓｌｓｙ，１９９３）の考え方を採用し、分類を図る。

【００３８】Ｃｏｌｌｅｃｔｉｏｎの種類として、図６に示すように、同一な要素を許す集合／許さない集合（Ｂａｇ／Ｓｅｔ）、さらにＢａｇの一種として順序付きの集合（Ｓｅｑｕｅｎｃｅ）、Ｓｅｔの一種としてｋｅｙ及びｖａｌｕｅの組を要素として持つ集合（Ｍａｐ）等がある。

【００３９】本発明では、異なる型を並べる複合型をＭａｐに分類し、同じ型を並べる複合型をＳｅｑｕｅｎｃ

ｅに分類する方法を提案する。Ｍａｐに分類するものを構造型、Ｓｅｑｕｅｎｃｅに分類するものを配列型と呼ぶことにする。

【００４０】この外、Ｃｏｌｌｅｃｔｉｏｎの概念に当てはめることができない、プロトコルに特徴的なデータ型がある。これは、ＡＳＮ．１におけるＡＮＹ（ＯｐｅｎＴｙｐｅ）、ＩＤＬにおけるａｎｙ等、実行時に決まる型である。この型については、実行時の型情報の取得設定インターフェースが必要であり、この型をＡＮＹ型と呼ぶことにする。

【００４１】以上を整理すると、表１の４つの型になる。

【００４２】

【表１】

型名	構造	説明
基本型	－	計算機言語の原始型であらわすことのできる型。
構造型	Map	複数の型を並べた型。
配列型	Sequence	同一の型を並べた型。可変長の配列。
ＡＮＹ型	特殊	任意の型を内包できる型。

【００４３】１．２．制約

データ構造をオブジェクトの集合として抽象化すると、従来のＡＰＩにあった型による開発者の誘導が全くできなくなる。それに代わる物として制約を導入する。制約は各データの型、取り得る値、要素数、要素の存在等の条件を表す情報でイベントオブジェクトから取得する。

【００４４】さらに制約の応用範囲を広げ、データ構造の一部を制約によって表す。例えば、ＡＳＮ．１のＣＨＯＩＣＥや、ＩＤＬのｕｎｉｏｎは、複数の型の候補のうち、１つを選択して使用するという複合型であるが、これらの型を、構造型に対して、要素のうち１つだけが存在するという制約を設けたものと解釈する。即ち、ＳＥＱＵＥＮＣＥとＣＨＯＩＣＥ、ｓｔｒｕｃｔとｕｎｉｏｎに対しては、構造上は同じインターフェースとし、制約によってその意味の違いを表す。このように、従来、構造として分類していた差異を制約の違いと解釈することによって、インターフェースの種類を縮小することができる。

【００４５】１．３．誘導情報・メタデータ

情報定義には、型名、構造型の要素の名前、数値につけられた名前等、実際に転送されるデータには含まれないが、開発者の理解を助ける情報が多く含まれている。ツールを用いて、的確に開発者を誘導するためには、これらの情報を制約情報等と共にツールがイベントオブジェクト自身から取り出せるようにしておく必要がある。

【００４６】一方、イベントオブジェクトは通常、実行時に生成される一時的なオブジェクトであり、コンポーネントの組み合わせ時にはインスタンスが存在しないことが考えられる。そのため、インターフェースに関する情報はメタデータとして１つのオブジェクトにまとめ、データ構造の名前等からメタデータを検索できるようにしておく必要がある。

２．インタフェース定義

以上の考え方に基づいたアクセスインターフェースの骨格をＪａｖａ（ＪＤＫ１．２）で定義した場合の例を以下に示す。ＪＤＫ１．２のＣｏｒｅ　ＡＰＩにはＣｏｌｌｅｃｔｉｏｎが導入され、アプリケーションに因らない共通なインターフェースとして推奨されている。ＪＤＫ１．２では、ＳｅｑｕｅｎｃｅをＬｉｓｔと呼んでいる。ＪＤＫのＣｏｌｌｅｃｔｉｏｎの重要な点は、インターフェースと実装を明確に分離しており、本発明のインターフェースをＣｏｒｅ　ＡＰＩに準備された汎用のライブラリと区別なく使えることである。

【００４７】また、同一Ｊａｖａ　Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅ内でデータを交換するインターフェースとして、Ｉｎｆｏ　Ｂｕｓ（）があり、原子型の取得設定にはこれを用いる。Ｉｎｆｏ　Ｂｕｓでは交換データのＡＰＩの１つとしてＣｏｌｌｅｃｔｉｏｎ　ＡＰＩを定めており、本発明のインターフェースはＩｎｆｏ　Ｂｕｓの仕様を満たしたものとなっている。

【００４８】２．１．データインターフェース

表２にデータアクセスのために使用するインターフェースを示す。新規に定義しているのは、Ｐｄｕ　Ｄａｔａ、Ｐｄｕ　Ａｎｙ　Ｄａｔａだけであり、それぞれ、表３、表４に示す。

【００４９】

【表２】

型	インターフェース	説明
基本型	PduData, ImmediateAccess	ImmediateAccessを介して基本型データを取得設定する
構造型	PduData, Map	Ｍａｐのｋｅｙには要素名を，ｖａｌｕｅには要素値を指定する
配列型	PduData, List	Ｌｉｓｔの要素に配列の要素を対応させる
ＡＮＹ型	PduData, PduAnyData	PduAnyDataで内包するデータ，メタデータを取得設定する

【００５０】　【表３】

戻り値	関数	説明
String	getTypeName（）	タイプ名取得
PduMetaData	getMetaName（）	メタデータ取得
Map	getRestriction（）	制約情報取得

【００５１】　【表４】

戻り値	関数	説明
void	setAnyMetaData（PduMetaData）	メタデータ設定
PduMetaData	getAnyMetaData（）	メタデータ取得
Void	getAnyData（PduData）	データ設定
PduData	getAnyData（）	データ取得

【００５２】２．２．メタデータインターフェース
型情報は制約の一種であるが、型はオブジェクトの生成削除に関わる基本構造であるため、制約情報から切り離してメタデータに組み込む。表５、表６にメタデータにアクセスするためのインタフェースを示す。

【００５３】　【表５】

型	インターフェース	説明
基本型	PduMetaData, ImmediateAccess	ImmediateAccessを介して基本型データのクラスを取得
構造型	PduMetaData, Map	Ｍａｐのｋｅｙに要素名を指定し，ｖａｌｕｅとして要素のメタデータまたはクラスを取得
配列型	PduMetaData ImmediateAccess	ImmediateAccessを介して要素のメタデータ・クラスを取得
ＡＮＹ型	PduMetaData	

【００５４】　【表６】

戻り値	関数	説明
int	getKind（）	基本型・構造型・配列型・ANY型の種別の取得
String	getProtocol（）	プロトコル名の取得
String	getTypeName（）	タイプ名の取得
PduMetaData	getParent（）	親のメタデータの取得
Map	getRestriction（）	制約情報の取得
PduData	newInstance（）	データの生成

【００５５】２．３．制約・誘導情報
データに与えられる制約の種類は無限にあり、一般的な記述方法はない。ＡＳＮ．１のＳｕｂ　ｔｙｐｅ及びＸＭＬのＤＴＤで表現できる制約を完全に表現できることを目標に、制約を整理すると表７になる。この表では、制約情報をＭａｐで表し、ｋｅｙが意味を、ｖａｌｕｅが値を表す。また、誘導のための情報を表８に表す。

【００５６】　【表７】

13　　14

key	value	型	説明
value	値，範囲の集合	全	取りうる値
size	整数値，整数範囲の集合	基本	文字列の長さ
		構造	存在する要素数
		配列	要素数
regexp	正規表現	基本	文字列の正規表現
present	要素名の集合	構造	存在すべき要素名
absent	要素名の集合	構造	存在しない要素名
type	型名の集合	ＡＮＹ	設定できるタイプ名
element	制約	基本	文字列の文字の制約
		構造	要素に共通の制約
named Element	要素名と制約	構造	特定の要素の制約
element Sequence	制約の並び	配列	要素の並びの制約
or	制約の集合	全	各制約の論理和
and	制約の集合	全	各制約の論理積
not	制約	全	制約の反転

【００５７】

【表８】

key	valueの型	型	説明
valueNames	名前と値のMap	全	各値に名前がある場合
indexNames	名前と整数値のMap	配列	配列型のインデックス値に名前がある場合
typeNames	名前とメタデータのMap	ＡＮＹ	各タイプに名前がある場合
defaultValue	値	全	デフォルト値
defaultValues	名前と値のMap	構造 配列	要素のデフォルト値。配列型の場合には，名前にindex値を用いる

【００５８】３．実装と評価

本発明のアクセスインターフェースについて、ＡＳＮ．１及びＸＭＬに対してＪａｖａにおける実装を行った一例を示す。ＡＳＮ．１とＸＭＬの表現力は似ており、１つの規格をＡＳＮ．１とＸＭＬの前身であるＳＧＭＬの双方で記述している場合がある。ところが、通常の実装では全く異なったＡＰＩが用いられるため、アクセスインターフェースの効果を試すのに最適な一例である。

【００５９】ＡＳＮ．１に対しては、Ｊａｖａ上に構築した独自の実装の上にアクセスインターフェースをかぶせる。ＸＭＬに対しては、ＸＭＬ及びＨＴＭＬのアクセスインターフェースとしてＷ３Ｃにより進められているＤＯＭ（ＤｏｃｕｍｅｎｔＯｂｊｅｃｔ　Ｍｏｄｅｌ）（Ｄｏｃｕｍｅｎｔ　Ｏｂｊｅｃｔ　Ｍｏｄｅｌ（ＤＯＭ）Ｌｅｖｅｌ　１　Ｓｐｅｃｉｆｉｃａｔｉｏｎ　Ｖｅｒ．１．０　Ｗ３Ｃ　Ｒｅｃ，１－Ｏｃｔ－１９９８．http://www.w3.org/TR/REC-DOM-Level-1）による実装の上にアクセスインターフェースをかぶせる。

【００６０】表９にＡＳＮ．１のタイプと本発明のインターフェースの対応を、表１０にＸＭＬの要素との対応を示す。

【００６１】

【表９】

15　　16

タイプ	型	制約・誘導情報
ENUMERATED	基本	value：namedNumber
INTEGER	基本	valueNames：namedNumber
SEQUENCE	構造	present：Optionalでない要素名 defaultValues：デフォルト値
CHOICE	構造	size：1
SET OF	配列	なし
BIT STRING	配列	element：Boolean indexNames：namedBit
OCTET STRING	配列	element：Byte
ANY	ＡＮＹ	なし
Subtype	親の型	size：SizeConstraint element：Permitted Alphabet namedElement：Inner subtyping等

【００６２】

【表１０】

タイプ	型	制約・誘導情報
タグ名	基本	values：上挿入可能なエレメント
CDATA	基本	regexp：]]>を含まない文字列
Document固有情報	構造	present：省略できない部分
Document配下のノード群	配列	なし
Document配下のノード	構造	size：1 absent：許されないノード種類
Element固有情報	構造	present：必須部（タグ名等）
Element配下のノード群	配列	elementSequence：エレメント定義の構造
Element配下のノード	構造	size：1 absent：許されないノード種類

【００６３】ＡＳＮ．１では、データ構造が予め厳格に定められているのに対し、ＸＭＬでは文章構造を表すＤＴＤの知識がなくともＷｅｌｌ－Ｆｏｒｍｅｄな文書としてこれを取り扱う必要がある。ＤＯＭでは文書の要素（エレメント、属性、テキスト等）をノードとしたツリーで構造を表しているが、１つのノードは、固有の情報（エレメントのタグ名等）と配下のノードを持っており、かつ、ＤＴＤに依存しないよう、配下のノードには任意の種類のノードが設定可能であるため、図７に示すように、１つのノードを３つのアクセスインターフェース（ノード固有情報を表す構造型、配下のノード群を表す配列型、配下のノードの種類を表す構造型）に分けて表すことにした。

【００６４】図８は本発明によるデータの設定のようすを示すもので、各コンポーネント２に対する各種の設定は前述したアクセスインターフェースに基づく共通インターフェース２１を介してツリー構造のメニュー２２で行われる。

【００６５】また、データ編集コンポーネントを製作し、ＡＳＮ．１、ＸＭＬとも同じコンポーネントで編集できることを確認した。この際、制約情報を適宜表示することによってある程度ユーザを誘導できることが分かった。ＡＳＮ．１、ＸＭＬの各編集画面の一例を、図９、図１０にそれぞれ示す。

４．制約のツール上における誘導への活用

請求項２、３に記載した本発明のＯｐＳの設計方法を、図１１、図１２にそれぞれ示す。

【００６６】従来は、図１３に示すようにエディタ上に置かれた２つのコンポーネント間の接続チェックに型のみを使用していた。しかし、コンポーネント間の接続はｓｙｎｔａｘな妥当性の上に、ｓｅｍａｎｔｉｃな妥当性が問われるはずであり、コンポーネント自身が持つ制約情報は、ｓｅｍａｎｔｉｃを完全とは言わないまでもかなりの部分を包含している。

【００６７】そこで、図１１では、２つのコンポーネント間を接続する際に、互いの制約条件を確認するようにする。そして、出力側の制約条件が、入力側の制約条件に比べて厳しい場合は、接続処理を行う。一方、そうでない場合は、それらのコンポーネント間の接続を拒絶する。

【００６８】さらに、図１２では、例え出力コンポーネント自身の制約条件が緩くても、そこを流れるデータの制約条件が厳しければ、接続を許すようにする。

【００６９】このように、図１１、図１２で説明した手順を従来のコンポーネント間接続のツールに挿入することにより、コンポーネント間でやり取りするデータの矛盾から生じるバグをＯｐＳの設計の段階で除去できる。

【００７０】

【実施の形態２】請求項４及び５に記載した本発明は、個々のネットワーク機器を接続した際における、機器

（ネットワークエレメント／ＮＥ）の設定に関係する。ここでは、ネットワークの管理対象となる機器（ＮＥ）に対し、ｈｔｔｐで制御し、そのｈｔｍｌ（またはＸＭＬ等）出力を管理システムが管理プロトコルデータとして扱い、それを走査して必要な部分の修正を請求項５に記載した発明で行い、その結果を管理システムのユーザーにｈｔｍｌとして送出することによって、個々の管理機器が提供する操作方法を最大限に活用するという請求項４の発明を説明する。

【００７１】図１４はこれらの発明の手法を用いている実施の形態の一例を示すもので、３１はネットワーク、３２はネットワークエレメント（ＮＥ）、３３は請求項４に記載した各手段、即ちネットワーク管理システムにおける管理対象となる機器のｈｔｔｐサーバが出力するＨＴＭＬファイルを、機器を管理するシステムがトラップする手段と、そのトラップしたＨＴＭＬファイルを、機器の現状における接続・設定の状態に照らして一部改変または最初から初期設定値を追加する等の処理を行うＨＴＭＬ処理手段と、処理後のＨＴＭＬを、そのネットワークシステムの管理者用のブラウザに転送する転送手段とを構成するマネージャ、３４はユーザが操作するＧＵＩ、例えばWWWブラウザを備えたＰＣである。

【００７２】ネットワーク３１はネットワークエレメント３２が複数接続されて構成されており、ネットワーク制御システム（ＯｐＳ）はこれらの機器を管理する。ここでは、管理対象となる各ネットワークエレメント３２はｈｔｔｐサーバを備えているとする。これらのネットワークエレメント３２は、ＳＮＭＰ等の一般的なネットワーク管理プロトコルによって一括的に管理することも可能であり、また、個々の機器に関しては、ＨＴＴＰプロトコルによって、ＧＵＩ３４を活用して、それらの機器にとって最善の形、即ちこれらの機器が提供するＧＵＩインターフェースを尊重する形で機器を管理することも可能である。

【００７３】ここで、個々のネットワーク機器を、ネットワーク管理システムのユーザが、ｈｔｔｐプロトコルを使って設定したいとする。すると、ユーザは、ｈｔｔｐリクエストを、マネージャ３３を経由して各機器に対して送出する。すると、この場合、マネージャ３３は、各管理対象機器が送出するｈｔｔｐの内容を最大限に尊重しつつ、それらの中でネットワークシステム全体に関する設定項目に対しては、マネージャ３３が最初から設定しておく形で、ユーザにｈｔｍｌまたはそれに類するデータを送るのが望ましい。そのため、一度送られてきたｈｔｍｌデータを、管理データとしてシステムの中で処理し、そのデータの中の必要な部分に対しては適宜予め定められた方法によって、データの設定等の処理を行い、その結果となるデータを、再びｈｔｍｌデータに直して、ユーザに送出する。

【００７４】このようにすると、ユーザは、各機器から送出されるであろう多数の設定項目のうち、予めネットワークシステム全体に関わる部分は最初から設定された形で設定項目を提示されるため、ネットワーク機器の設定が簡易化し、設定の負担が軽くなり、また、設定の誤りによってネットワーク全体に障害が及ぶのを防ぐことが可能になる。これが、請求項４で述べた発明である。

【００７５】次に、これをコンポーネント指向の枠組みで実現する手段として、請求項５で述べられた方法をどのように適用するかを説明する。

【００７６】請求項５の発明は、図１５に示すように、イベントの通知が巨大なデータオブジェクト４１で伝わってきた際に、それをより細かい複数のデータオブジェクト４２単位に走査する処理を行うコンポーネント４３である。

【００７７】図１５の左側からオブジェクト４１がイベントとして次段のコンポーネントへ伝わっている。このイベント通知で、データを複数のより細かい部分に走査するコンポーネント４３が、入力された巨大なデータオブジェクト４１をより細かいデータオブジェクト４２として出力する。細分化することで、次段のコンポーネントは小さなデータオブジェクト単位に処理することが可能になる。その処理結果は、必要ならば、もとに反映させることが可能となる。

【００７８】ここで断っておくが、従来の通信プロトコルデータは、全て１つのＰＤＵに収まることを仮定していた。しかしながら、ＨＴＭＬの構造型データやＣＭＩＰのＬｉｎｋｅｄ　Ｒｅｐｌｙ、ＡＳＮ．１のｓｅｔｏｆ等、個々の要素の操作をコンポーネントで行いたい場合は、このような従来の手法をそのまま適用することは困難である。そのため、図１５で説明した複数データオブジェクトに分割するコンポーネントの採用等、何らかの形でデータオブジェクトデータを細切れにした形でも処理ができるようなコンポーネントを考えて準備しておくことが重要である。

【００７９】ここで、図１６を用いて、請求項４の発明を、請求項５の発明でどのように実現するかを具体的に説明する。まず、ＮＥ５１から、ＨＴＴＰプロトコルリクエストの返答として、ＨＴＴＰプロトコルレスポンスによって、ＨＴＭＬデータがマネージャ５２に返ってくるところから考える。この時、マネージャ５２内のＨＴＴＰ受信コンポーネント５３は、ＨＴＭＬデータを他の通信プロトコルのデータと同様に、コンポーネント間でやりとりされる形のツリー上の構造データに変換して、それを出力先のコンポーネントに送り出す。ここで、このＨＴＴＰ受信コンポーネント５３は出力データを、請求項５で述べた、データ走査コンポーネント５４に入力として渡すとする。

【００８０】データ走査コンポーネント５４は、渡されたツリー構造型のデータを、個々に走査（分解）して、個々のデータとして、例えばＨＴＭＬを調べて適宜必要

な設定項目について設定を行うコンポーネントに渡す。ＦＯＲＭ設定処理コンポーネント５５は、次々と細切れになって送られてくる各入力データについて逐次調べ、それがカスタマイザで指定されたタイプのＦＯＲＭデータであれば、それを必要に応じて変更する。このようにして、大きい構造型データの各部分について、処理を行うことが可能になる。

【００８１】このようにして処理されたデータを、データ走査コンポーネント５４は、マネージャ５２を操作するユーザへのＨＴＴＰ（送信）コンポーネント５６に送出する。ＨＴＴＰコンポーネント５６は、受け取った構造型データを再びＨＴＭＬに直して、ユーザのＧＵＩ５７に送る。

【００８２】こうすることによって、ユーザはＮＥ５１から送られてくるデータのうち、マネージャ５２が必要に応じて修正した部分を除いて、ほぼＮＥ５１が提供するＧＵＩをそのまま活用する形で利用することが可能になるのである。

【００８３】

【実施の形態３】従来のコンポーネント技術によるＯｐＳ構築手法では、設計時には、サーバ・クライアントや、マネージャ・エージェントのような、運用時に１：ｎの動的な関係になるようなコンポーネントの接続関係でも、１：１の静的に接続されていた。しかしながら、実際には、これらマネージャ・エージェントの接続は常に設計当初のままの１：１の静的な関係ということはなく、場合によっては、１：ｎの関係となる。これには、エージェントの増設（新設したエージェントを同じマネージャに接続し追加する）や、逆に一部を撤去（稼働していたエージェントをマネージャから切り離す）するような場合も含まれる。同様の動的な接続・切り放しの関係は、サーバ・クライアント間でも起りうる。

【００８４】ツールで接続したら、その場ですぐにその接続を実行したり、シミュレートしたりできるのが、コンポーネント指向の特徴であった。しかしながら、このような１：ｎの関係は、ツール上でシミュレートするのは困難であった。その解決方法として、１：ｎの分界点となるコンポーネントが、自身でｎ個側のインスタンスをカスタマイザの設定に応じて生成したり、または生成先に対して適切に送出したりすることによって、この関係をシミュレートする。

【００８５】この仕組みは、請求項６記載の発明のＯｐＳ設計方法では、図１７に示すように、コンポーネントをマネージャとエージェントとの間にまたがる２つのコンポーネントに分離するようにする。ＯｐＳ設計ツール６０上ではコンポーネント６１から６５までが接続されている。このうち、ＯｐＳ設計の当初からコンポーネント６１はクライアントで、６５はＮＥである。また、コンポーネント６２，６４はネットワーク上での通信を示すコンポーネントである。

【００８６】実際のＯｐＳが運用される時には、マネージャ（またはサーバ）６６にコンポーネント６２，６３，６４が移され、さらに、そのコンポーネント６２，６４は分離（６２’，６４’）してクライアントとエージェントにも移される。

【００８７】こうすることで、クライアント側やエージェント側が変わったり、複数のクライアントやエージェントがサーバまたはマネージャと接続し合っても、対応可能となる。図１８はＮＥを動的に登録した後の結合関係を示すものである。

【００８８】本発明では、運用時における、このようなネットワーク上での１：ｎの関係を、設計時に仮想的にコンポーネント６１，６５のインスタンスをコンポーネント６２や６４が増やし、また、コンポーネント６２や６４がコンポーネント６３からのデータを適切に振り分けたり、その逆を行うことによって、この状況を適切にシミュレーションできることを示す。

【００８９】多様な主信号を扱うマルチメディアネットワークでは、様々な管理プロトコルが使用される。本発明では通信網管理システムをコンポーネント指向で製作することを提案し、異なる種類のプロトコルデータを同一のコンポーネントで取り扱うためのアクセスインターフェースを示した。また、Ｃｏｌｌｅｃｔｉｏｎの概念と制約情報を用いて、プロトコル種別に依存しないインターフェースを定義し、ＡＳＮ．１やＸＭＬ等の異なるプロトコルのデータに対し、それらの構造を維持したまま、汎用的なコンポーネントがデータの処理を維持できることを示した。本発明のインターフェースは、プロトコル固有のＡＰＩが全く異なる両者に適用可能で、同一のデータ編集コンポーネントで双方の操作が可能である。

【００９０】実施の形態では、本発明による通信網管理システムをコンポーネント指向で製作する場合に必要な、プロトコルデータをコンポーネント間で交換するためのインターフェースについて述べた。Ｃｏｌｌｅｃｔｉｏｎの概念と制約情報を用いて、プロトコル種別に依存しないインターフェースを定義し、ＡＳＮ．１とＸＭＬに対して実際にＪａｖａを用いてインプリメントを行う。両者のプロトコル固有のＡＰＩは全く異なっているが、本発明のインターフェースは双方に適用可能である。本発明のインターフェースを介してデータの編集を行うＧＵＩコンポーネントは双方に適用でき、プロトコルに依存することなくデータを操作できる。

【００９１】

【発明の効果】以上説明したように、請求項１の発明によれば、多様なデータ構造を同時に統一されたデータアクセスインターフェースでアクセスすることを、ツリー型エディタで実現できる。この結果、ユーザがＯｐＳを設計するのに必要となるコンポーネントウェア（ＣＷ）を設定することが容易となり、ＣＷ間でやり取りされる

21

データも視覚的な理解が可能になる。

【００９２】また、請求項２，３の発明によれば、コンポーネント間での接続を適切に誘導するために、コンポーネントがやりとりできるデータに対して制約を導入し、この制約に基づいて、コンポーネント間の接続の可否（誘導）を決めた。このようにすることで、ＯｐＳ設計で誤った接続を解消でき、設計ミスを削減できる。

【００９３】また、請求項４記載の発明によれば、多数のネットワーク機器が接続されている場合、その中の１つの機器の設定変更を行うＧＵＩについては、個々の機器が搭載している設定用ＧＵＩ／ｈｔｔｐサーバを活用し、変更に伴うネットワーク全体の調整を図るためにマネージャを活用することが可能になる。

【００９４】また、請求項５記載の発明によれば、１つのデータオブジェクトに対して、その構造の中身について個々の処理をコンポーネントが行いたい場合、そのデータオブジェクトに対して走査・分解を行うコンポーネントを導入し、その使用によって、例えば請求項２，３の発明をコンポーネント指向の枠組みのなかで行うことを示した。

【００９５】さらにまた、請求項６記載の発明によれば、１：ｎの分界点におけるコンポーネントを境にして、分界点からｎ側のコンポーネントブロックのインスタンスを、シミュレート時に、分界点のプロトコルを表すコンポーネントが多数作成・初期化する。また、それらを分界点コンポーネントが適切にデータの流れの振り分け処理をする。このようにすることによって、ツール上でも、ネットワークの１：ｎ関係を適切にシミュレートすることを可能にした。

【図面の簡単な説明】

【図１】ネットワーク制御システムの代表的な構成図

【図２】従来のコンポーネント型のネットワーク制御システムの設計の流れを示す図

【図３】ＯｐＳの設計におけるデータの設定のようすを示す図

【図４】ネットワークエレメントに対する操作のようすを示す図

【図５】本発明のアクセスインターフェースの概要を示す図

【図６】複合型のデータ構造として用いるＣｏｌｌｅｃｔｉｏｎの種類を示す図

【図７】ＤＯＭとアクセスインターフェースとの対応を示す図

【図８】本発明によるデータの設定のようすを示す図

【図９】ＡＳＮ．１データの編集画面の一例を示す図

【図１０】ＸＭＬデータの編集画面の一例を示す図

【図１１】請求項２対応のコンポーネント間の接続の処理フロー

【図１２】請求項３対応のコンポーネント間の接続の処理フロー

【図１３】従来のコンポーネント間の接続の処理フロー

【図１４】請求項４、５にかかわる実施の形態の一例を示す図

【図１５】巨大なデータオブジェクトの処理のようすを示す図

【図１６】請求項５にかかわる処理の一例を示す図

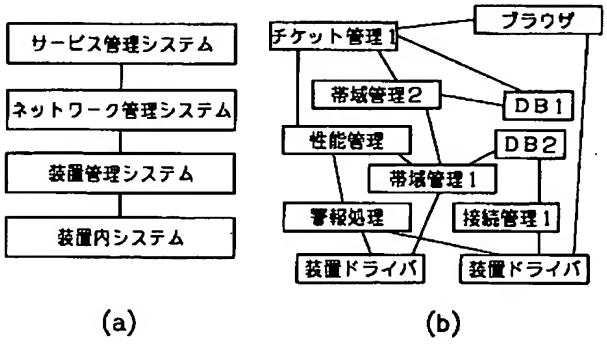
【図１７】請求項６対応の実施の形態の一例にかかわるＮＥの動的な登録のようすを示す図

【図１８】請求項６対応の実施の形態の一例にかかわる登録した後の結合関係を示す図

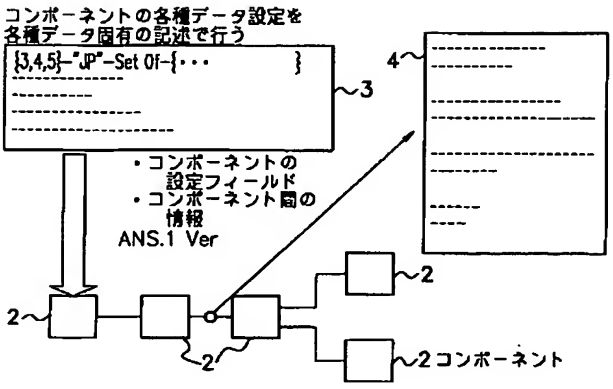
【符号の説明】

２，１４，１５：コンポーネント、１１：固有データ、１２：アクセスインターフェース、１３：イベントオブジェクト、２１：共通インターフェース。

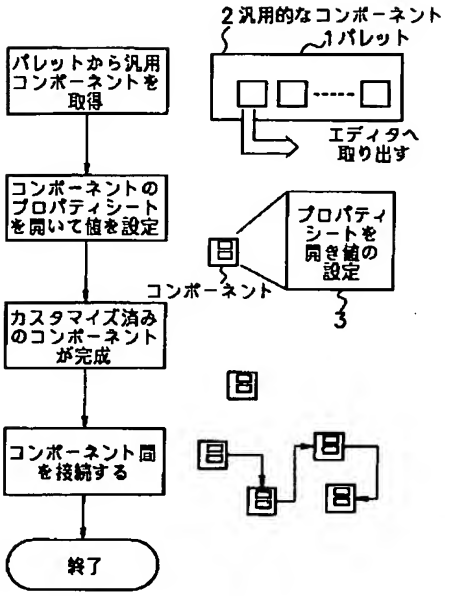
【図１】



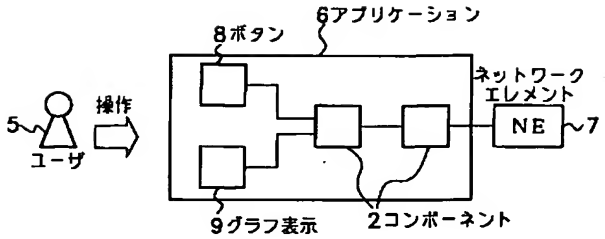
【図３】



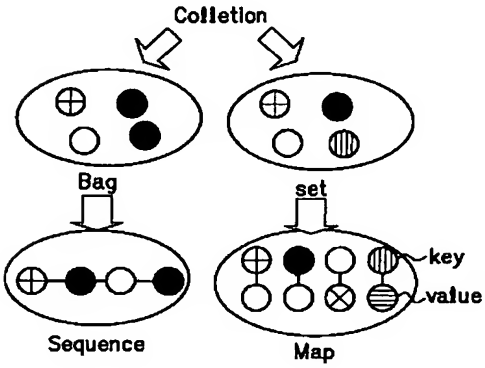
【図２】



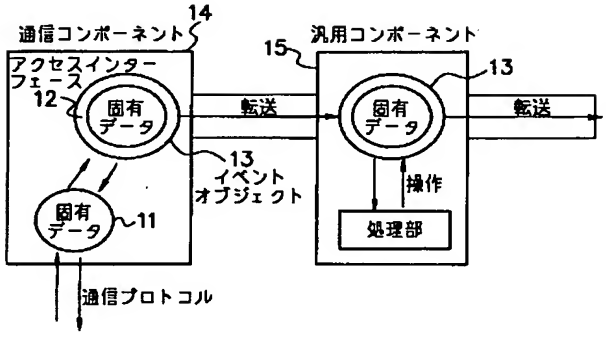
【図４】



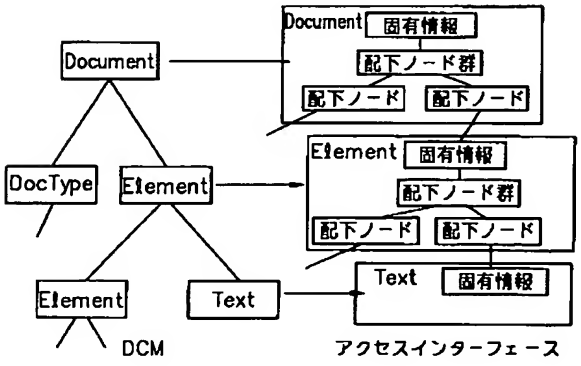
【図６】



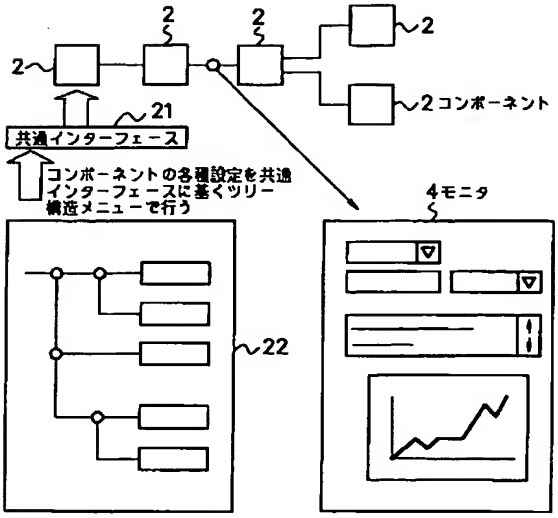
【図５】



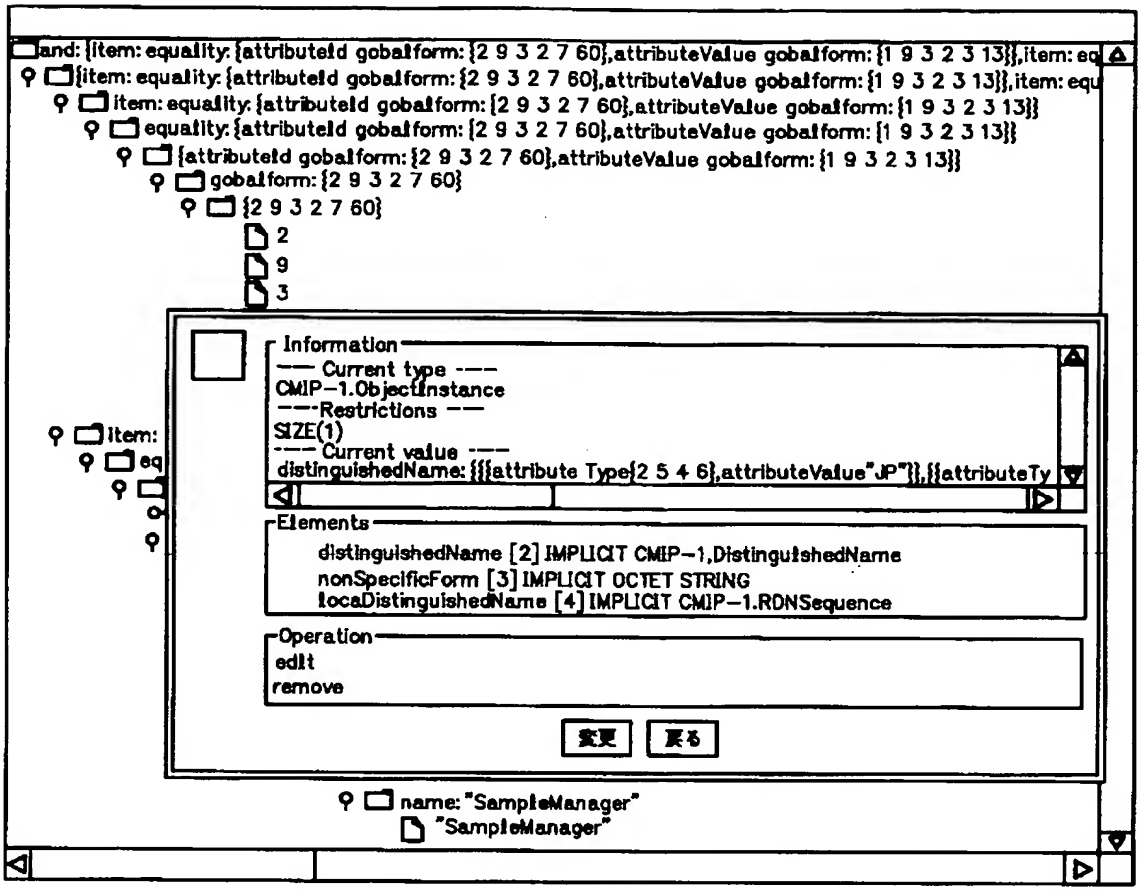
【図７】



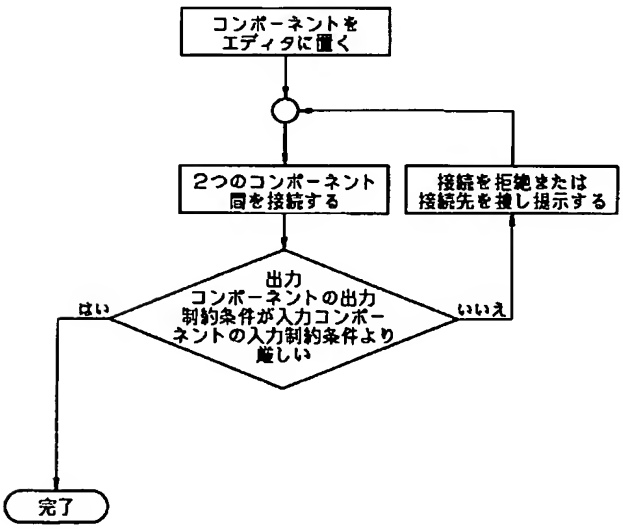
【図８】



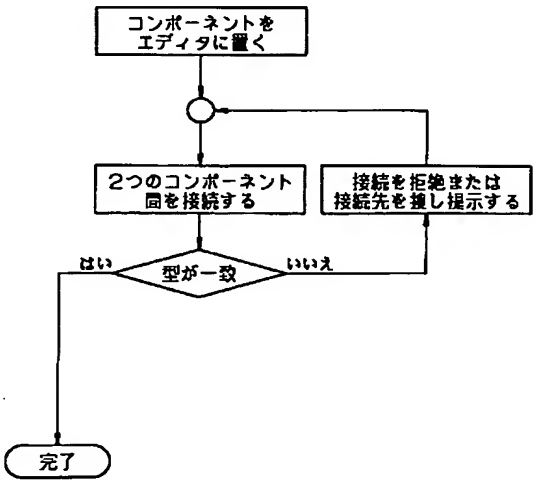
【図９】


and: {item: equality: {attributeId gobalform: {2 9 3 2 7 60},attributeValue gobalform: {1 9 3 2 3 13}},item: eq
{item: equality: {attributeId gobalform: {2 9 3 2 7 60},attributeValue gobalform: {1 9 3 2 3 13}},item: equ
item: equality: {attributeId gobalform: {2 9 3 2 7 60},attributeValue gobalform: {1 9 3 2 3 13}}
equality: {attributeId gobalform: {2 9 3 2 7 60},attributeValue gobalform: {1 9 3 2 3 13}}
{attributeId gobalform: {2 9 3 2 7 60},attributeValue gobalform: {1 9 3 2 3 13}}
gobalform: {2 9 3 2 7 60}
{2 9 3 2 7 60}
2
9
3
Information
--- Current type ---
CMIP-1.0ObjectInstance
---Restrictions ---
SIZE(1)
--- Current value ---
distinguishedName: {{{attribute Type{2 5 4 6},attributeValue"JP"}},{{attributeTy
Elements
distinguishedName [2] IMPLICIT CMIP-1.DistinguishedName
nonSpecificForm [3] IMPLICIT OCTET STRING
locaDistinguishedName [4] IMPLICIT CMIP-1.RDNSequence
Operation
edit
remove
変更
戻る
item:
eq
name: "SampleManager"
"SampleManager"

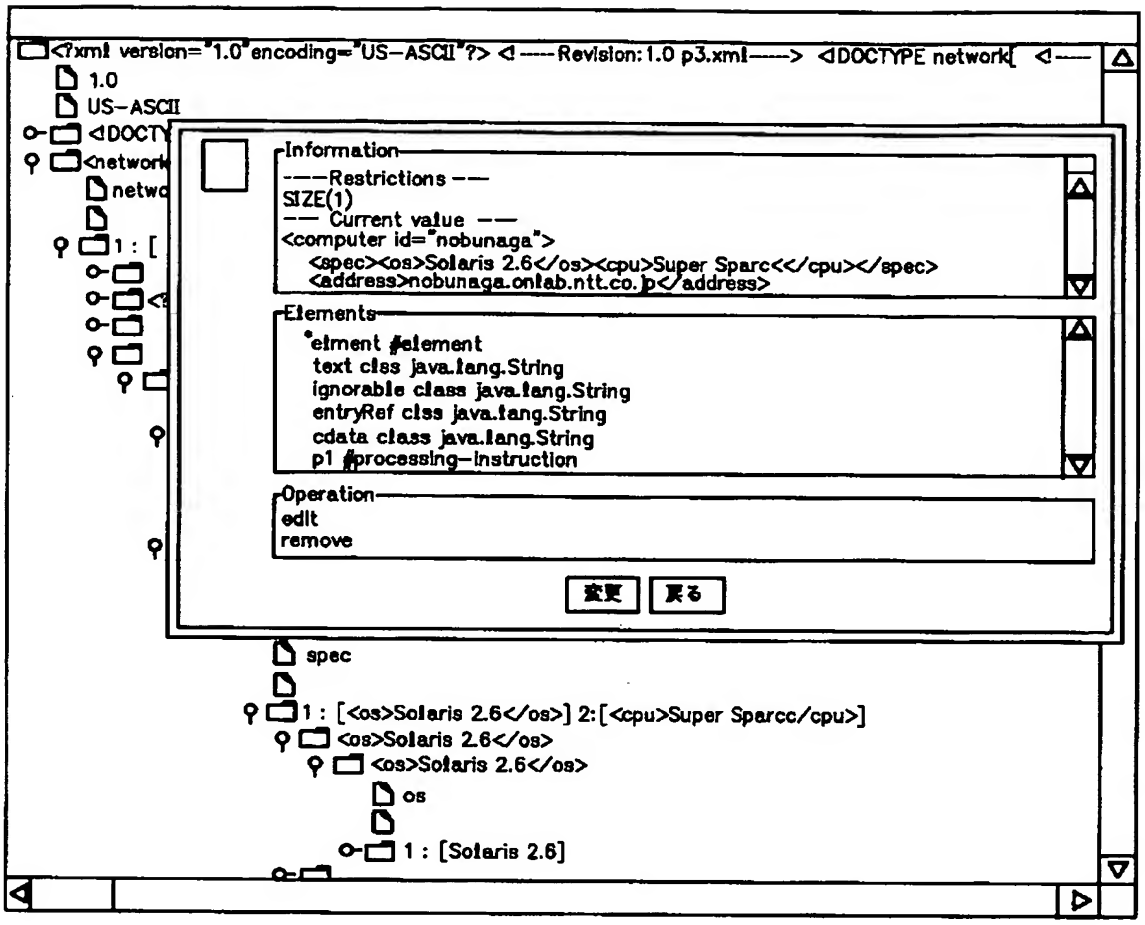
【図１１】


コンポーネントを
エディタに置く
2つのコンポーネント
同を接続する
接続を拒絶または
接続先を捜し提示する
出力
コンポーネントの出力
制約条件が入力コンポー
ネントの入力制約条件より
厳しい
はい
いいえ
完了

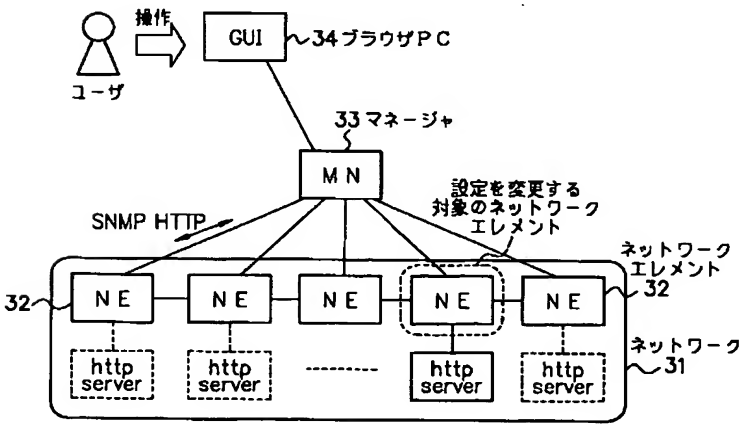
【図１３】


コンポーネントを
エディタに置く
2つのコンポーネント
同を接続する
接続を拒絶または
接続先を捜し提示する
はい
型が一致
いいえ
完了

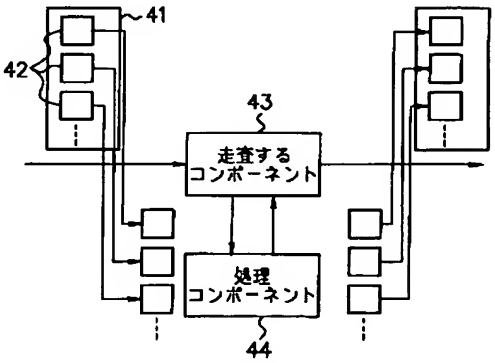
【図１０】


<?xml version="1.0" encoding="US-ASCII"?> <!---- Revision: 1.0 p3.xml----> <!DOCTYPE network[<!----
1.0
US-ASCII
Information
----Restrictions ----
SIZE(1)
---- Current value ----
<computer id="nobunaga">
<spec><os>Solaris 2.6</os><cpu>Super Sparc</cpu></spec>
<address>nobunaga.onlab.ntt.co.jp</address>
Elements
element #element
text clss java.lang.String
ignorable class java.lang.String
entryRef clss java.lang.String
cdata class java.lang.String
p1 #processing-instruction
Operation
edit
remove
変更
戻る
spec
1 : [<os>Solaris 2.6</os>] 2:[<cpu>Super Sparcc/cpu>]
<os>Solaris 2.6</os>
<os>Solaris 2.6</os>
os
1 : [Solaris 2.6]

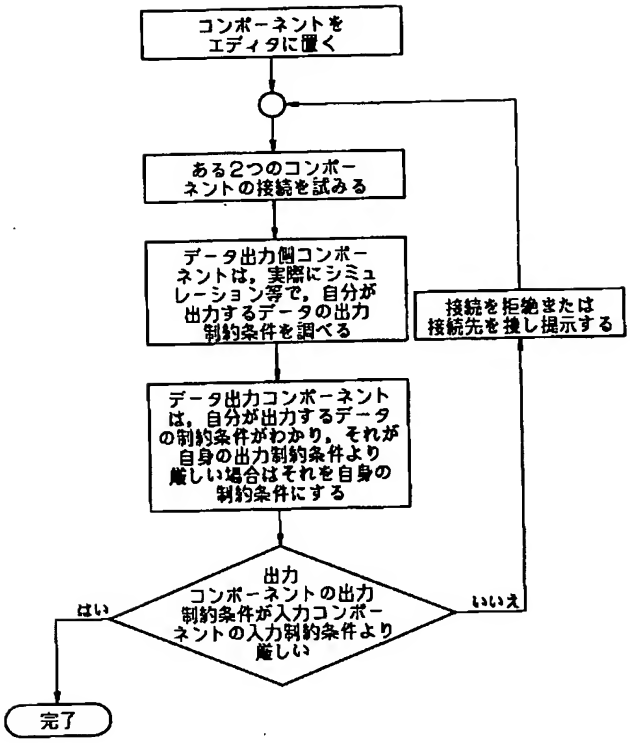
【図１４】


操作
GUI
34 ブラウザＰＣ
ユーザ
33 マネージャ
ＭＮ
設定を変更する
対象のネットワーク
エレメント
SNMP HTTP
ネットワーク
エレメント
32
ＮＥ
ＮＥ
ＮＥ
ＮＥ
ＮＥ
32
http server
http server
http server
http server
ネットワーク
31

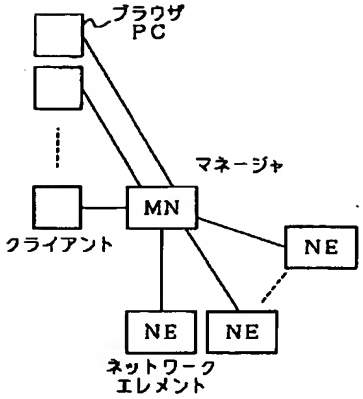
【図１５】


41
42
43
走査する
コンポーネント
処理
コンポーネント
44

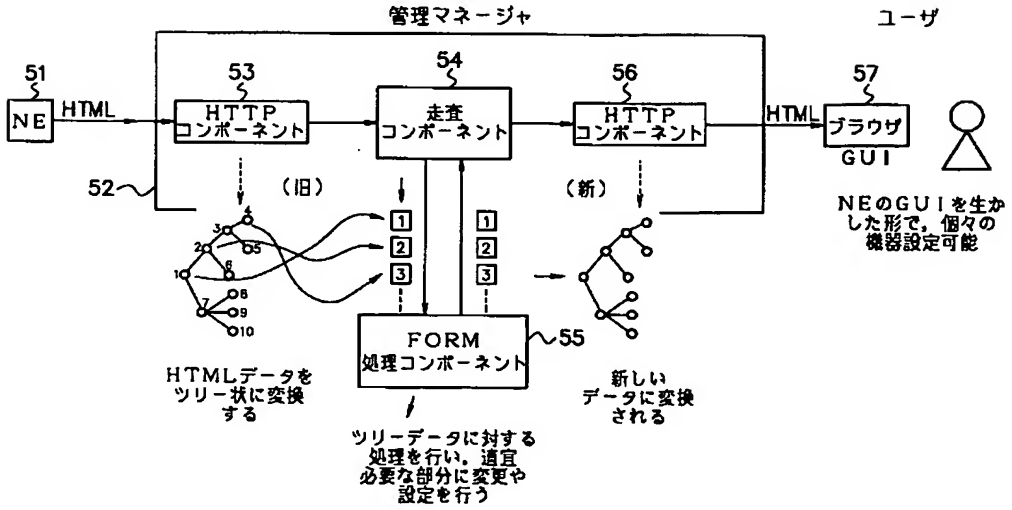
【図１２】


コンポーネントを
エディタに置く
ある2つのコンポー
ネントの接続を試みる
データ出力側コンポー
ネントは，実際にシミュ
レーション等で，自分が
出力するデータの出力
制約条件を調べる
接続を拒絶または
接続先を提し提示する
データ出力コンポーネント
は，自分が出力するデータ
の制約条件がわかり，それが
自身の出力制約条件より
厳しい場合はそれを自身の
制約条件にする
出力
コンポーネントの出力
制約条件が入力コンポー
ネントの入力制約条件より
厳しい
はい
いいえ
完了

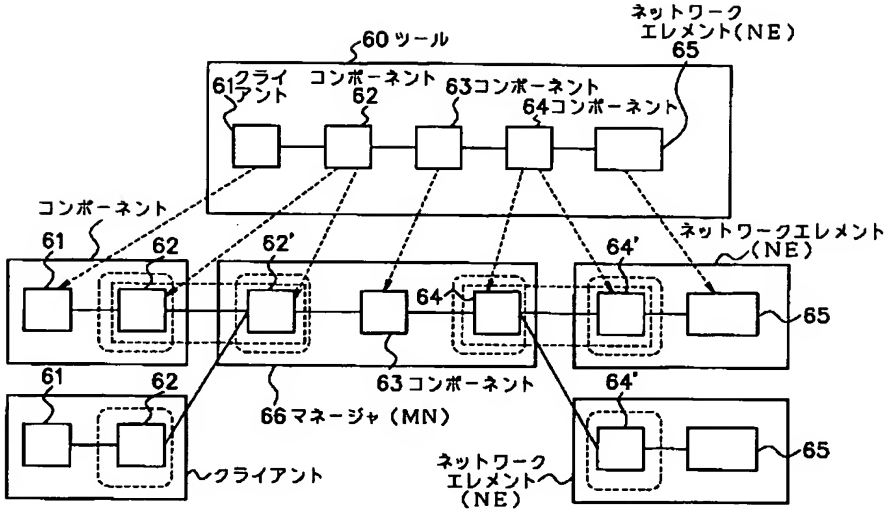
【図１８】


ブラウザ
PC
マネージャ
MN
クライアント
NE
NE
NE
ネットワーク
エレメント

【図１６】


管理マネージャ
ユーザ
51
NE
HTML
53
HTTP
コンポーネント
54
走査
コンポーネント
56
HTTP
コンポーネント
HTML
57
ブラウザ
GUI
NEのGUIを生か
した形で，個々の
機器設定可能
52
（旧）
（新）
1
2
3
FORM
処理コンポーネント
55
HTMLデータを
ツリー状に変換
する
新しい
データに変換
される
ツリーデータに対する
処理を行い，適宜
必要な部分に変更や
設定を行う

【図１７】



フロントページの続き

Ｆターム(参考)　5B076 DD05
5K034 AA06 AA18 FF01 FF11 FF15
FF18 HH14 HH17 HH18 HH26
JJ23 MM39